IN THE CLAIMS

1. [currently amended] A process comprising:

validating a ~~formal language specification~~ Formal Language Specification written in a formal language which has predetermined rules of syntax and semantics, said ~~formal language specification~~Formal Language Specification defining a computer program to be automatically written;

automatically translating each element of a ~~formal language specification~~ Formal Language Specification defining an object model, a functional model, a dynamic model and a presentation model, which taken together define the requirements of the program to be automatically written, into a full and complete computer program which needs no additional third party source code or source code from existing components or code libraries to be compiled with it to make said computer program complete and which implements the requirements of said ~~formal language specification~~ Formal Language Specification, said ~~formal language specification~~ Formal Language Specification defining at least an identification function for every class, and at least a valuation for every variable attribute, said translating step comprising the following steps:

using a computer, automatically write computer code that will request user name and password, receive any responses and authenticate the user;

using a computer, automatically write computer code that has the capability to control a computer to provide a mechanism to ~~will~~ determine a ~~this~~ user's privilege level from log in information supplied by said user which identifies said user and query said ~~formal language specification~~ Formal Language Specification and determine all object attributes said user has privilege to see and all services said user has privileges to ~~can~~ invoke;

using a computer, automatically write computer code which has the capability to query ~~queries~~ said ~~formal language specification~~Formal Language Specification for all services of all classes that any authorized user has privileges to ~~may~~ invoke and identify ~~identifies~~ an object server which will implement each said service;

using a computer, automatically write code that has the capability to ~~will~~ retrieve service arguments for all services;

using a computer, automatically write code that is capable of controlling ~~controls~~ a computer to provide a ~~display~~ means by which an entity has a

mechanism to ~~and entity can~~ invoke a service, and which has a mechanism to receive ~~receives~~ input to invoke a particular service and respond ~~responds~~ by sending a message to the appropriate object server to invoke the service, said message including the ~~necessary~~ arguments required for the service to execute;

using a computer, automatically write code that has the capablity to control a computer to implement ~~implements~~ an object server for every service, each of which first checks to verify that state transitions are valid ~~and make sense~~ for the current state of objects of which the object server will be altering the state;

using a computer, automatically write code that has the capability to control a computer to implement ~~for~~ every object server that verifies preconditions are satisfied before making state transitions of any objects the states of which are acted upon by the object server;

using a computer, automatically write code that has the capability to control a computer to make all valuation calculations required by said ~~formal language specification~~ Formal Language Specification of each object server;

using a computer, automatically write code that has the capability to control a computer to verify that integrity constraints specified in said ~~formal language specification~~ Formal Language Specification on the values of attributes of objects have been satisfied after execution of a service and respond by reversing any changes in state which caused said integrity constraint to be violated ~~take action~~ if said integrity constraints are not satisfied; and

using a computer, automatically write code that has the capability to control a computer to implement ~~for~~ every object server such that said object server tests ~~to test~~ trigger relationships specified in said ~~formal language specification~~ Formal Language Specification after execution of a service and invoke a predetermined service associated with a trigger event ~~carry out appropriate action~~ if said ~~a~~ trigger event has occurred.

2. [currently amended] An apparatus for automatically translating a ~~formal language specification~~Formal Language Specification defining an object model, a functional model, a dynamic model and a presentation model, which taken together define the requirements of a computer program to be automatically written, said ~~formal language~~

~~specification~~Formal Language Specification being written in a formal language which has predefined rules of grammar, said translating acting to convert said ~~formal language specification~~Formal Language Specification into a computer program that implements the requirements of said ~~formal language specification~~Formal Language Specification, said ~~formal language specification~~Formal Language Specification defining at least an identification function for every class and at least a valuation for every variable attribute, said apparatus comprising:

a computer programmed with an operating system and one or more other programs to cooperate with said operating system to control said computer to perform the following functions:

A) using said predetermined rules of grammar to validate said ~~formal language specification~~Formal Language Specification to ensure that said ~~formal language specification~~Formal Language Specification is complete and correct;

B) automatically write computer code that will request user name and password, receive any responses and authenticate the user;

C) automatically write computer code that will determine a user's privilege level for a user identified by user name and password entered in response to the query caused by the code written in step B, and query said formal specification and determine all object attributes said user has privilege to see and query and all services said user has privileges to ~~can~~ invoke;

D) automatically write computer code which queries said formal specification for all services of all classes that any authorized user has privileges to ~~may~~ invoke and identifies an object server which will implement said service;

E) automatically write code that will retrieve service arguments for all services;

F) automatically write code that displays one or more user interface tools which provide a mechanism ~~can be used~~ to invoke a service, and which provides a mechanism to receive ~~receives~~ input to invoke a particular service and which responds by sending a message to the appropriate object server to invoke said service, said message including the ~~necessary~~ arguments required for said service to execute;

G) automatically write code that implements an object server for every service, each of which first checks to verify that state transitions are valid ~~and~~

~~make sense~~ for the current state of objects the object service will be altering the state of;

H) automatically write code for every object server that verifies preconditions are satisfied before making state transitions of any objects the states of which are acted upon by said object server;

I) automatically write code to make all valuation calculations required by said specification of each object server;

J) automatically write code to verify that integrity constraints specified in said Formal Language Specification ~~formal specification~~ on the values of attributes of objects have been satisfied after execution of a service and reversing any changes in state which caused said integrity constraints to be not satisfied ~~take action~~ if said integrity constraints are not satisfied; and

K) automatically write code for every object server to test trigger relationships specified in said Formal Language Specification ~~formal specification~~ after execution of a service and invoke a predetermined service associated with a trigger event ~~carry out appropriate action~~ if a trigger event has occurred.

3. [currently amended] A physical computer-readable storage media ~~medium~~ containing instructions for controlling a computer to automatically translate a ~~formal language specification~~Formal Language Specification defining an object model, a functional model, a dynamic model and a presentation model, which taken together define the requirements of a computer program to be automatically written, said ~~formal language specification~~Formal Language Specification defining at least an identification function for every class and at least a valuation for every variable attribute said ~~formal language specification~~Formal Language Specification written in a formal language having predefined rules of grammar, by:

validating a ~~formal language specification~~Formal Language Specification written in a formal language which has predetermined rules of syntax and semantics, said validating accomplished using said predetermined rules of syntax and semantics to ensure said ~~formal language specification~~Formal Language Specification is complete and correct;

automatically writing computer code that will request user name and password, receive any responses and authenticate the user;

automatically writing computer code that will determine a user's privilege level and query said Formal Language Specification ~~formal specification~~ and determine all object attributes said user has privilege to see and all services said user has privileges to ~~can~~ invoke;

automatically writing computer code which queries said Formal Language Specification ~~formal specification~~ for all services of all classes that any authorized user has privileges to ~~may~~ invoke and identifies an object server which will implement said service;

automatically writing computer code that will retrieve service arguments for all services;

automatically write code that displays menus options, icons or creates any other means or mechanism through ~~by~~ which a user or another process has the capability to ~~can~~ invoke a service, and which provides a mechanism through which ~~receives~~ input to invoke a particular service is provided and mechanisms to provide values for arguments is provided and a mechanism is provided to construct and send ~~responds by sending~~ a message to the appropriate object server to invoke the service, said message including the ~~necessary~~ arguments required for the service to execute;

automatically writing code that implements an object server for every service, each of which first checks to verify that state transitions are valid ~~and make sense~~ for the current state of objects the object service will be altering the state of;

automatically write code for every object server that verifies preconditions are satisfied before making state transitions of any objects the states of which are acted upon by the object server;

automatically write code to make all valuation calculations required by said Formal Language Specification ~~formal specification~~ of each object server;

automatically write code to verify that integrity constraints specified in said Formal Language Specification ~~formal specification~~ on the values of attributes of objects have been satisfied after execution of a service and reverse any state changes which have been made which cause said integrity constraints to be not satisfied ~~take action~~ if said integrity constraints are not satisfied; and

automatically write code for every object server to test trigger relationships specified in said Formal Language Specification ~~formal specification~~ after execution of a service and invoke a predetermined service associated with a trigger event ~~carry out~~

~~appropriate action~~ if a trigger event has occurred.

4. [currently amended] An apparatus for automatically translating a Formal Language Specification written in any formal language defining a full and complete Conceptual Model of a desired computer program to be automatically generated into a full and complete source code which implements said desired computer program, comprising:

a computer programmed with an operating system and one or more other programs to cooperate with said operating system to control said computer to perform the following functions:

reading all said primitives in said Formal Language Specification in any order;

in any order, using a computer and said Formal Language Specification to automatically generate computer code which has the capability to control a computer to carry out ~~generating~~ one or more methods ~~comprised of computer code which can control a computer to perform the following functions~~ in an order determined by an execution model:

determining if said Formal Language Specification requires user authentication, and, if so, automatically writing computer code that will request user name and password, receive any responses and authenticate the user;

determining if said Formal Language Specification requires determining a user privilege level, and, if so, automatically writing computer code that will determine a user's privilege level and query said Formal Language Specification and determine all object attributes said user has privilege to see and determine all services this user has privileges to ~~can~~ invoke;

determining if said Formal Language Specification defines services, and, if so, automatically writing computer code which queries said Formal Language Specification to determine all services that the authenticated user has privileges to ~~may~~ invoke and which are defined in said Formal Language Specification for all classes of objects said authenticated user will be able to view and automatically writing an object

server for each said service which will implement said service upon receipt of a service invocation message, each of said object servers containing code which will perform the following functions in the following order upon receipt of a service invocation message:

verify that one or more proposed state transitions are valid ~~can be validly made~~ for the current state of any object(s) of which said object server will be altering the state before actually altering the state of said object(s);

verify that any preconditions of a service implemented by an object server are satisfied before said object server will act upon said one or more objects to make state transitions thereof in carrying out said service,

ignoring said service invocation message if either said one or more state transitions cannot be validly made for the current state of any objects upon which said object server will be acting or any said precondition is not satisfied;

if all said proposed transitions are valid ~~can be validly made~~ on said one or more objects upon which said object server will act and if all said preconditions are satisfied, make all valuation calculations of said object server required by said Formal Language Specification;

verify that service execution by said object server did not result in violation of one or more integrity constraints specified in said Formal Language Specification on the values of attributes of objects affected by execution of said service implemented by said object server, and take corrective action if one or more of said integrity constraints are not satisfied; and

after a valid change of state of an object acted upon by said object server occurs, test trigger relationships or condition-action rules specified in said Formal Language Specification and, if any trigger event is satisfied, triggering a service specified in said condition-action rule or trigger relationship.

5. [Cancelled]

6. [Cancelled]

7. [Cancelled]

8. [Cancelled]

9. [Cancelled]

10. [Currently amended] A process for automatically translating a Formal Language Specification written in any formal language defining a full and complete Conceptual Model of a desired computer program to be automatically generated into a full and complete source code which implements said desired computer program, comprising:

reading all said primitives in said Formal Language Specification in any order;

in any order, using a computer and said Formal Language Specification to automatically generate computer code that has the capability to control a computer to implement ~~generating~~ one or more methods which perform the following functions in an order determined by an execution model:

determining if said Formal Language Specification requires user authentication, and, if so, automatically writing computer code that will request user name and password, receive any responses and authenticate the user;

determining if said Formal Language Specification requires determining a user privilege level, and, if so, automatically writing computer code that will determine a user's privilege level and query said Formal Language Specification and determine all object attributes said user has privilege to see and determine all services this user has privileges to ~~can~~ invoke;

determining if said Formal Language Specification defines services, and, if so, automatically writing computer code which queries said Formal Language Specification to determine all services that the authenticated user has privileges to ~~may~~ invoke and which are defined in said Formal Language Specification for all classes of objects said authenticated user will be able to view and automatically writing an object server for each said service which will implement said service upon receipt of a service invocation message, each of said object server

containing code which will perform the following functions in the following order upon receipt of a service invocation message:

verify that one or more proposed state transitions are valid ~~can be validly made~~ for the current state of any object(s) of which said object server will be altering the state before actually altering the state of said object(s);

verify that any preconditions of a service implemented by an object server are satisfied before said object server will act upon said one or more objects to make state transitions thereof in carrying out said service,

ignoring said service invocation message if either said one or more state transitions cannot be validly made for the current state of any objects upon which said object server will be acting or any said precondition is not satisfied;

if all said proposed transitions are valid ~~can be validly made~~ on said one or more objects upon which said object server will act and if all said preconditions are satisfied, make all valuation calculations of said object server required by said Formal Language Specification;

verify that service execution by said object server did not result in violation of one or more integrity constraints specified in said Formal Language Specification on the values of attributes of objects affected by execution of said service implemented by said object server, and take corrective action if one or more of said integrity constraints are not satisfied; and

after a valid change of state of an object acted upon by said object server occurs, test trigger relationships or condition-action rules specified in said Formal Language Specification and, if any trigger event is satisfied, triggering a service specified in said condition-action rule or trigger relationship.

11. [Cancelled]

12. [Cancelled]

13. [Previously Presented] A process for converting a Formal Language

Specification encoding a Conceptual Model which defines desired system logic and a desired user interface of a desired computer program into source code which encodes said desired computer program, comprising:

validating said Formal Language Specification to ensure it is complete and correct;

retrieving predetermined information from said Formal Language Specification encoding a Conceptual Model which defines one or more classes of objects in an object model, a dynamic model which specifies the behavior of each object in response to services, triggers and global transactions as represented by a state transition diagram for every class and an object interaction diagram for every trigger and for every global transaction, and a functional model which defines the semantics of any change of each object's state as a consequence of an event occurrence by specifying for each class one or more mathematical or logical formulas which define how one or more variable attributes of said class will have their values changed when one or more specified events of said class occurs meaning one or more services of said class is executed;

using predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a presentation tier of said desired computer program;

using predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a persistence tier, database of data structure of said desired computer program;

using said predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a middle tier of said desired program which communicates with said presentation tier in the manner defined below, said middle tier written by automatically generating at least the following component instances for each class defined in said Formal Language Specification:

a server component instance including a method to implement each service present in a signature of said class and one or more methods to receive requests from said presentation tier that relate to execution of services of said class;

a query component instance including a method for implementing queries to extract information from said persistence tier relating to objects within said class and a method to receive and process requests from said presentation tier to query said persistence tier;

an executive component instance including one or more methods to receive and process a request from said server component or another executive component to execute a service in said class and carry out the following functions:

verify the existence and validity for a requested server component;

create a copy of a requested server component instance in memory and access said persistence tier using said query component to retrieve values of constant and variable attributes of said server component;

validate state transitions for said requested service and a present state for a server component instance;

verify the satisfaction of preconditions specified in said Formal Language Specification of said requested service;

changing the state of said server component instance to a new state by modifying a value of a variable attribute of said server component instance by performing all valuations specified in said functional model affected by said requested service;

validating said new state by verifying said new state does not violate static or dynamic restrictions specified in said Formal Language Specification;

check trigger conditions established in said Formal Language Specification to determine if said new state causes any trigger to occur, and, if so, which actions should be carried out;

communicate with said persistence tier to access or store all attributes of said server component instance.

14. [currently amended] A process for validating a ~~formal language specification~~Formal Language Specification, comprising the steps:

A) checking said ~~formal language specification~~Formal Language Specification to ensure that it is complete in that all required properties of a Conceptual Model embodied in said ~~formal language specification~~Formal Language Specification are defined and have a valid value;

B) using rules of grammar of whatever formal language said ~~formal language specification~~Formal Language Specification is written in, checking said ~~formal language specification~~Formal Language Specification to ensure it is correct in that it is syntactically and semantically correct and not ambiguous.

15. [Previously Presented] The process of claim 14 further comprising the step of presenting a request for said missing information via a mechanism of a user interface if any information is missing or presenting a request via a user interface mechanism to correct any syntactic or semantic error or clarify any ambiguity discovered during said validation process.

16. [Currently Amended] The process of claim 14 further comprising the step of doing a partial validation each time an element is added to said ~~formal language specification~~Formal Language Specification to check for completeness and correctness and mark the portion of said ~~formal language specification~~Formal Language Specification just added as invalid if an error is found so that a request to correct said error is ~~can be~~ presented later when a full validation of said ~~formal language specification~~Formal Language Specification is requested.

17. [Previously Presented] The process of claim 14 wherein step A comprises checking to ensure that all elements in said Conceptual Model have a set of properties that exist and have a valid value and wherein step B comprises using a predetermined process and grammar for every type of formula in said Conceptual Model to ensure each is syntactically and semantically correct.

18. [Previously Presented] The process of claim 14 wherein step A comprises performing strict validation on some element properties and flexible validation of other element properties of said Conceptual Model, said strict validation of a property defined as requiring a full definition and valid value for a property, and flexible validation defined as allowing a property to be incomplete or have an invalid value during a process of inputting elements which define said Conceptual Model, and wherein the following table defines which elements and properties have strict or flexible validations:

Element	Property	Subproperty	Validation Type
class			
	name		strict
	ID function		flexible
	attributes (at least one)		flexible
	services (at least a Create service)		flexible
	static and dynamic integrity constraints	their formulas	strict
attribute			
	name		strict
	type (constant, variable, derived)		strict
	data-type (real, integer, etc.)		strict
	default value		strict
	size		strict
	request in creation service		strict
	null value allowed		strict
	evaluations (variable attributes)		flexible
	derivation formula (derived attributes)		flexible

Evaluation			
	one variable attribute of a class		strict
	one service of the same class		strict
	condition		strict
	formula of evaluation		strict
derivation			
	formula		strict
	condition (optional)		strict
service			
	name		strict
	arguments		
		argument's name	strict
		data type	strict
		default value (optional)	strict
		null value	strict
		size (if proceeds)	strict
		formula of transaction	flexible
preconditions of an action			
	formula		strict

		agents affected by condition	strict
relationship: aggregation			
	related classes (component and composite)		strict
	relationship name		strict
	both directions: role names		strict
	cardinality		strict
	inclusive or referential		strict
	dynamic		strict
	clause "group by" (optional)		strict
	insertion and deletion events (if proceed)		strict
relationship: inheritance			
	related classes (parent & child)		strict
	temporal (versus permanent)		strict
	specialization condition or events		strict

relationship: agent			
	agent class and service allowed to activate		strict
state transition diagram			
	all states of classes (three at least)		flexible
state in state transition diagram			
	name		strict
transition in state transition diagram			
	estate of origin		strict
	estate of destination		strict
	service of class		strict
		control condition (optional)	strict
trigger			
	condition		strict
	class or instance of destination		strict
	target (self, object, class)		strict
	activated service		strict

	service arguments' initialization (optional)		
		argument values	strict
global interactions			
	name		strict
	formula		strict
user exit functions			
	name		strict
	return data-type		strict
	arguments, (optional)		
	argument's name		strict
	argument's data-type		strict.

19. [Previously Presented] The process of claim 14 wherein step B comprises validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas, and wherein a predetermined process for validation and a set of rules of grammar exist for each type of formula and a validation process and a set of rules appropriate for the type of formula being validated is used for validation of each formula.

20. [Currently Amended] The process of claim 14 wherein step B comprises validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas defined by the table below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments
inheritance: specialization conditions	
static and dynamic integrity constraints	
derivations and valuations	
	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	
local and global transactions formulas	

and further comprising the steps of:

presenting dialog boxes via a user interface by which a user is provided a mechanism to ~~may~~ enter said formulas during a process of defining said Conceptual Model;

and wherein step B is performed by preventing a user from leaving a dialog box being used to define a formula until said formula being defined is syntactically and semantically correct.

21. [Previously Presented] The process of claim 14 wherein steps A and B are performed by at least checking to ensure that every formula is syntactically correct, every class has an identification function and has a creation event and a destroy event, every triggering formula is semantically correct, every name of an aggregation is unique in the scheme of said Conceptual Model, every derived attribute has at least a derivation formula, every service has an agent or server declared to execute it.

22. [Currently Amended] The process of claim 14 further comprising performing steps A and B each time a user working on defining said Conceptual Model makes a change which may invalidate one or more formulas, but wherein predetermined formulas are allowed to be temporarily incorrect so that the user is presented them for ~~can~~ review ~~them at~~ a later time if they are still incorrect when a full validation process is performed after the user is done defining the Conceptual Model.

23. [Currently Amended] The process of claim 14 further comprising the step of using a computer to automatically translate said ~~formal language specification~~Formal Language Specification into computer code after steps A and B and been completed and all formulas are syntactically and semantically complete and correct.

24. [Currently Amended] The process of claim 14 further comprising the steps:

presenting user interface tools by which a user may define said Conceptual Model and make changes thereto;

checking all affected formulas each time a change is made to said Conceptual Model;

if the change affects a strictly validated property, then the change is rejected if the property is not given a valid value, otherwise the change is accepted;

if the change affects a property which is not strictly validated, then the user is informed should any error arise, but allowed to do the modification if said user ~~he or she~~ wishes;

if there are no affected formulas, modifying the Conceptual Model as specified by the user.

25. [Currently Amended] The process of claim 14 wherein user interface defining portions of said ~~formal language specification~~Formal Language Specification are validated by:

verifying that any patterns defined by said user are acceptable user interface patterns with no essential information missing;

attributes used in filters specified as part of said user interface are visible

from a definition class;

attributes used in order criteria are visible from a definition class;

any formula in a filter is syntactically and semantically correct and uses only terms defined in said Conceptual Model;

any action selection pattern uses as final actions object defined in said Conceptual Model;

any set of dependency patterns are terminal and have confluence; and warnings are displayed to said user if any pattern is defined but not used or if an instance pattern is duplicated.

26. [Currently Amended] An apparatus comprising a computer programmed with an operating system and a validation program that cooperates with said operating system to control said computer to perform the following functions of a validation process:

A) check a ~~formal language specification~~Formal Language Specification for completeness by checking to ensure said ~~formal language specification~~Formal Language Specification has no missing information which is needed to detail the requirements of a desired computer program modelled by said ~~formal language specification~~Formal Language Specification;

B~~b~~) cooperate with said operating system to ensure said ~~formal language specification~~Formal Language Specification is correct by checking each statement in said ~~formal language specification~~Formal Language Specification according to rules of grammar of whatever formal language in which said ~~formal language specification~~Formal Language Specification is written to ensure that each statement is syntactically and semantically correct and not ambiguous so as to ensure that all properties of elements in a Conceptual Model encoded in said ~~formal language specification~~Formal Language Specification have a value which is valid and to ensure that all formulas in said Conceptual Model have correct syntax and meaning.

27. [Previously Presented] The apparatus of claim 26 wherein said validation program controls said computer to perform the following additional function:

presenting a request for said missing information via a mechanism of a user interface if any information is missing or presenting a request via a user

interface mechanism to correct any syntactic or semantic error or clarify any ambiguity discovered during said validation process.

28. [Currently Amended] The apparatus of claim 26 wherein said validation program controls said computer to perform the following additional function:

doing a partial validation each time an element is added to said ~~formal language specification~~Formal Language Specification to check for completeness and correctness and mark the portion of said ~~formal language specification~~Formal Language Specification just added as invalid if an error is found so that a request to correct said error is ~~can be~~ presented later when a full validation of said ~~formal language specification~~Formal Language Specification is requested if said error still exists.

29. [Previously Presented] The apparatus of claim 26 wherein said validation program controls said computer to perform the following additional function:

checking to ensure that all elements in said Conceptual Model have a set of properties that exist and have a valid value;

performing step B by using a predetermined process and grammar for every type of formula in said Conceptual Model to ensure each is syntactically and semantically correct.

30. [Previously Presented] The apparatus of claim 26 wherein said validation program controls said computer to perform step B by validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas, and wherein a predetermined process for validation and a set of rules of grammar exist for each type of formula and a validation process and a set of rules appropriate for the type of formula being validated is used for validation of each formula.

31. [Previously Presented] The apparatus of claim 26 wherein said validation program controls said computer to perform step B by validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas defined by the table

below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments
inheritance: specialization conditions	
static and dynamic integrity constraints	
derivations and valuations	
	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	
local and global transactions formulas	

and wherein said validation program is further structured to control said computer to perform the steps of:

presenting dialog boxes via a user interface by which a user may enter said formulas during a process of defining said Conceptual Model;

and wherein step B is performed by preventing a user from leaving a dialog box being used to define a formula until said formula being defined is syntactically and semantically correct.

32. [Currently Amended] The apparatus of claim 26 wherein said validation program controls said computer to perform the following steps:

presenting user interface tools by which a user may define said Conceptual Model and make changes thereto;

checking all affected formulas each time a change is made to said

Conceptual Model;

if the change affects a strictly validated property, then the change is rejected if the property is not given a valid value, otherwise the change is accepted;

if the change affects a property which is not strictly validated, then the user is informed should any error arise, but allowed to do the modification if said user ~~he or she~~ wishes;

if there are no affected formulas, modifying the Conceptual Model as specified by the user.

33. [Currently Amended] ~~An~~ A physical computer readable storage media ~~medium~~ having stored thereon computer-readable instructions which when executed by a computer implement a validation program to control said computer to perform the following functions of a validation process:

A) check a ~~formal language specification~~Formal Language Specification for completeness by checking to ensure said ~~formal language specification~~Formal Language Specification has no missing information which is needed to detail the requirements of a desired computer program modelled by said ~~formal language specification~~Formal Language Specification;

B~~b~~) cooperate with said operating system to ensure said ~~formal language specification~~Formal Language Specification is correct by checking each statement in said ~~formal language specification~~Formal Language Specification according to rules of grammar of whatever formal language in which said ~~formal language specification~~Formal Language Specification is written to ensure that each statement is syntactically and semantically correct and not ambiguous so as to ensure that all properties of elements in a Conceptual Model encoded in said ~~formal language specification~~Formal Language Specification have a value which is valid and to ensure that all formulas in said Conceptual Model have correct syntax and meaning.

34. [Previously Presented] The computer-readable medium of claim 33 wherein said validation program is further structured to control said computer to perform the following additional function:

presenting a request for said missing information via a mechanism of a user interface if any information is missing or presenting a request via a user interface mechanism to correct any syntactic or semantic error or clarify any ambiguity discovered during said validation process.

35. [Currently Amended] The computer-readable medium of claim 33 wherein said validation program is further structured to control said computer to perform the following additional function:

doing a partial validation each time an element is added to said ~~formal language specification~~Formal Language Specification to check for completeness and correctness and mark the portion of said ~~formal language specification~~Formal Language Specification just added as invalid if an error is found so that a request to correct said error is ~~can be~~ presented later when a full validation of said ~~formal language specification~~Formal Language Specification is requested if said error still exists.

36. [Previously Presented] The computer-readable medium of claim 33 wherein said validation program is further structured to control said computer to perform the following additional functions:

checking to ensure that all elements in said Conceptual Model have a set of properties that exist and have a valid value;

performing step B by using a predetermined process and grammar for every type of formula in said Conceptual Model to ensure each is syntactically and semantically correct.

37. [Previously Presented] The computer-readable medium of claim 29 wherein said validation program is structured to control said computer to perform step B by validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas, and wherein a predetermined process for validation and a set of rules of grammar exist for each type of formula and a validation process and a set of rules appropriate for the type of formula being validated is used for validation of each formula.

38. [Previously Presented] The computer-readable medium of claim 33 wherein said validation program is structured to control said computer to perform step B by validating formulas to ensure each formula complies with a precise syntax defined for a formula of that type and is semantically correct, where there are several types of formulas defined by the table below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments
inheritance: specialization conditions	
static and dynamic integrity constraints	
derivations and valuations	
	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	
local and global transactions formulas	

and wherein said validation program is further structured to control said computer to perform the steps of:

presenting dialog boxes via a user interface by which a user may enter said formulas during a process of defining said Conceptual Model;

and wherein step B is performed by preventing a user from leaving a dialog box being used to define a formula until said formula being defined is syntactically and semantically correct.

39. [Currently Amended] The computer-readable medium of claim 33 wherein said

validation program is structured to control said computer to perform the following steps:

presenting user interface tools by which a user may define said Conceptual Model and make changes thereto;

checking all affected formulas each time a change is made to said Conceptual Model;

if the change affects a strictly validated property, then the change is rejected if the property is not given a valid value, otherwise the change is accepted;

if the change affects a property which is not strictly validated, then the user is informed should any error arise, but allowed to do the modification if said user ~~he or she~~ wishes;

if there are no affected formulas, modifying the Conceptual Model as specified by the user.

40. [New] A process to automatically translate a Formal Language Specification defining the functionality of a computer application modelled in a Conceptual Model, into a computer program called an application, said process comprising the steps of:

A) validating said Formal Language Specification to ensure said Formal Language Specification is complete in that there is no missing information in said Formal Language Specification and to ensure said Formal Language Specification is correct in that primitives of said conceptual model are syntactically and semantically consistent and not ambiguous;

B) translating said validated Formal Language Specification into computer readable code which has the capability to control a computer to provide a user interface access mechanism to allow users to log in by entering at least identification data and to use said identification data to authenticate and validate a user as an instance of a class of the validated fFormal Language Specification that act as agent in at least one agent relationship;

C) translating said validated Formal Language Specification into computer readable code which has the capability to control a computer to provide a view of the system defining the set of objects and attributes the user can query and the set of services said user can execute, the content of said system view

depending on the identity of said user accessing said application;
and

D) translating said validated Formal Language Specification into computer readable code which has the capability to control a computer to provide user interface interaction mechanisms to interact with and execute the functionality of the application in terms of performing queries on information managed by said application and executing services to modify the state of said information managed by said application, said services comprising events, local transactions and global transactions.

41. [New] The process of claim 40 wherein Step D comprises translating said Formal Language Specification into computer code which has the capability to control a computer so as to implement functionality of said queries as defined in said validated Formal Language Specification by means of filter formulas of filter patterns representing said queries, and so as to implement functionality of said services as defined in said validated Formal Language Specification in terms of:

- at least a valuation for each variable attribute of each class in said validated Formal Language Specification associated to at least an event of said class, which altogether define the functionality of every event;
- a transaction formula that defines a composition of services into a molecular execution unit, thereby defining functionality of every local transaction and global transaction;
- state transitions to control the valid lives for objects of each class in said validated Formal Language Specification, upon occurrence of an event or a local transaction;
- optional preconditions to the execution of services; and
- optional integrity constraints to prevent the execution of services from leaving the information managed by said application in an inconsistent or invalid state.

42. [New] The process of claim 40 wherein said step B creates computer readable code which has the capability to control a computer to provide user interface access mechanisms which block access such that only users that belong to any class of said view for which said computer application is produced can connect or log on to

said desired computer program, said control being performed by requesting that a user wishing to log onto said desired computer program indicate information that identifies a class of which said user is an instance, and also indicate information that identifies the user as an instance of said class so as to identify said user, and also indicate information that is used as a password for that user so as to authenticate said user.

43. [New] The process of claim 40 wherein said step C creates computer readable code which has the capability to control a computer to provide said view of the system in such a manner that said system view comprises user interaction mechanisms included in an Action Selection Presentation Pattern associated with said view for which the application is produced.

44. [New] The process of claim 43 wherein step C creates computer readable code which has the capability to control a computer to restrict the user interface interaction mechanisms to the ones the user who logged on is allowed to interact with according to privileges established by data structures which relate the class said user belongs to, playing the role of agent, with classes, playing the role of servers, that determine which services of each server class will be available for execution by said user and which attributes of each server class said user will be able to query.

45. [New] The process of claim 40 wherein step D comprises translating said validated Formal Language Specification into computer readable code which has the capability to control a computer to provide user interface interaction mechanisms which allow a user who has logged on to interact with and access functionality of said computer application by invoking services, executing queries and/or execution of user interface interaction scenarios.

46. [New] The process of claim 45 wherein step D comprises the steps of translating said Formal Language Specification into computer readable code which has the capability to control a computer to provide user interface interaction mechanisms which allow a user who has logged in to interact with and access the functionality of said computer application by execution of only selected services selected from a group of services comprising events, local transactions and/or global transactions and

depending upon the identity of said user.

47. [New] The process of claim 46 where the steps in step D of translating said Formal Language Specification to generate computer readable code which controls a computer to allow a logged on user to interact with and execute the functionality associated to an event comprise generating code which controls a computer to provide a mechanism to construct the message associated to said event and to execute said event.

48. [New] The process of claim 47 wherein said step D includes the steps of translating said validated Formal Language Specification to generate computer readable code which controls a computer to provide a mechanism to construct said message associated to said event comprises generating code to control a computer to perform the following steps:

- providing a mechanism to identify the object on which the event will be executed, except if said event is a creation event, in which case this step will be omitted;
- providing mechanisms to give a value to every argument of said event, said mechanisms further controlling that arguments that require a value are provided a value and that any argument whose value is provided receives a valid value.

49. [New] The process of claim 47 wherein said step D includes the steps of translating said validated Formal Language Specification to generate computer readable code which has the capability to control a computer to provide a mechanism to execute a service which is an event and comprises the following steps:

- translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to, except in the case of a creation event, verify the existence of said object on which said event will be executed, or the non-existence of the object to be created in the case of a creation event;
- translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to, except in the

case of a creation event, recover the state of the object on which the event is executed from whatever memory or database or repository or any other persistence means (hereafter just "memory") to which said state of said object has been saved;

- translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to verify that, according to the state transition diagram of the class owning said event, there is a valid state transition labelled with said event being executed and for the agent class to which the user belongs who logged onto said desired computer program, in which case said mechanism will update the state of the object on which the event is executed according to said state transition diagram, and if there is no valid state transition, said mechanism will produce an error message causing the execution of the event to stop and roll back all changes made to the state of the object on which said event is executed

- translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to verify that every precondition that is defined for said event being executed and for said agent class to which the user logged on to the computer application belongs, is satisfied, and should any of said preconditions not be satisfied, then said mechanism will produce an error with the error message being predefined for said precondition that does not hold causing the execution of the event to stop and roll back all changes made to the state of said object on which said event is executed;

- except in the case of creation events or destruction events, translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to produce the changes of values to the variable attributes of said class owning said event for which valuation formulas in said functional model have been defined such that said valuations relate said variable attributes with the event being executed, and wherein said mechanism applies only the changes to the variable attributes which are required by valuation formulas of valuations whose valuation condition formula evaluates to

true, if any, or the change required by a valuation formula of a valuation having no valuation condition formula;

- in the case of a creation event, translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to assign a value to every constant or variable attribute of an object on which said creation event is executed and establishing relationships between said object on which said creation event is executed with objects of classes related with the class owning said creation event;
- in the case of a destruction event, translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to delete relationships of the object on which said destruction event is executed with objects said object is related to;
- except in the case of destruction events, translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to check that every integrity constraint defined in the class owning said event is satisfied, for the object whose state has been changed by said event, and should any of said integrity constraints not be satisfied, said mechanism will produce an error with the error message predefined for said integrity constraint that does not hold, said mechanism causing the execution of the event to stop and roll back all changes made to the state of the object on which the event is executed;
- except in the case of a destruction event, translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to save the changes made to said object on which the event is executed to memory;
- in the case of a destruction event, translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to delete the object on which said destruction event is executed from memory to which said object has been saved;

- except in the case of a destruction event, translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to check every trigger condition on said object on which said event is executed for every trigger relationship defined on said class owning said event, and wherein for every trigger condition that holds, a mechanism to execute the service associated to said trigger relationship by:
    - providing a mechanism to determine the set of objects on which said service associated to said trigger will be executed, and
    - executing said service on every object of said set of objects by:
        - giving a value to every argument of said service, and ensuring that every argument that requires a value is provided a value and that any argument whose value is provided receives a valid value; and
        - invoking the execution of said service on said object;

  and
- translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to inform the requestor of the result of executing the event.

50. [NEW] The process of claim 46 where the steps in step D of translating said Formal Language Specification to generate computer readable code which controls a computer to allow a logged on user to interact with and execute the functionality associated to a local transaction comprise generating code which controls a computer to provide a mechanism to construct the message associated to said local transaction and to execute said local transaction.

51. [New] The process of claim 50 wherein the steps in step D of translating said Formal Language Specification to generate computer readable code which controls a computer to provide a mechanism to construct said message associated to the local transaction comprise generating code to control a computer to perform the following

steps:

- providing a mechanism to identify the object on which said local transaction will be primarily executed, except if said transaction is a creation service, in which case this step will be omitted;
- providing mechanisms to give a value to every argument of said local transaction, said mechanisms further controlling that arguments that require a value are provided a value and that any argument whose value is provided receives a valid value.

52. [New] The process of claim 50 wherein said step D includes the steps of translating said validated Formal Language Specification to generate computer readable code which has the capability to control a computer to execute a service which is a local transaction and comprises the following steps:

- translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to, except in the case of a creation service, verify the existence of the object on which the local transaction will be executed or, in the case of a creation service, verify the non-existence of the object to be created;
- translating said validated Formal Language Specification so as to provide code which controls a computer to, except in the case of a creation service, provide a mechanism to recover the state of the object on which the local transaction is executed from memory to which the state of said object has been saved;
- translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to verify that, according to the state transition diagram of the class owning said local transaction, there is a valid state transition labelled with said local transaction being executed and for the agent class to which the user logged on to the system belongs, and, if there is such a valid state transition, said mechanism will update the state of the object on which the local transaction is executed according to said state transition diagram, and, if there is no such valid state transition, said mechanism will produce an error message causing the execution of said local transaction to stop and roll back all changes made to the state of said object

on which the local transaction is executed and of any other object the state of which has been modified by the execution of said local transaction;

- translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to verify that every precondition that is defined for the local transaction being executed and for the agent class to which the user logged on to the system belongs, is satisfied, and should any of said preconditions not hold, then said mechanism will produce an error with the error message defined for said precondition that does not hold causing the execution of said local transaction to stop and roll back all changes made to the state of the object on which said local transaction is executed and of any other object the state of which has been modified by the execution of said local transaction;
- for every service comprised in the transaction formula of said local transaction being executed:
  - o if said service has an associated guard, translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to check that said guard associated to said service holds; and if said guard does not hold, the rest of the steps associated to said service will be omitted;
  - o translating said validated Formal Language Specification so as to provide code which controls a computer to provide a mechanism to determine the set of objects on which said service comprised in said transaction formula will be executed and to provide a mechanism to execute said service on every object of said set of objects by:
    - providing mechanisms to give a value to every argument of said service, said mechanisms further ensuring that arguments that require a value are provided a value and that any argument whose value is provided receives a valid value;
    - providing a mechanism to invoke the execution of said service on said object and control the result of said execution, and should said execution result in an error, causing the execution of the local transaction to stop and roll back all changes made to the state of said object on which the local transaction is executed and of any

other object the state of which has been modified by the execution of said local transaction

- translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to check that every integrity constraint holds which is defined in the class owning the local transaction and in classes whose instances include objects the state of which has been modified by the execution of said local transaction, for any object whose state has been changed by said local transaction, and, should any of said integrity constraints not hold, said mechanism produces an error with an error message defined for said integrity constraint that does not hold and causing the execution of said local transaction to stop and roll back all changes made to the state of the object on which the local transaction is executed and of any other object the state of which has been modified by the execution of said local transaction;
- translating said validated Formal Language Specification so as to provide code which has the capability to control a computer to provide a mechanism to check every trigger condition on any object the state of which has been modified by the execution of said local transaction, for every trigger relationship defined on each class owning an object the state of which has been changed by execution of said local transaction, and for every trigger condition that holds, a mechanism to execute the service associated to said trigger relationship by:
    - providing a mechanism to determine the set of objects on which said service associated to said trigger will be executed, and
    - providing a mechanism to execute said service on every object of said set of objects by:
        - giving a value to every argument of said service, and ensuring that every argument that requires a value is provided a value and that any argument whose value is provided receives a valid value; and
        - invoking the execution of said service on said object;

and

- translating said validated Formal Language Specification so as to provide code

which has the capability to control a computer to provide a mechanism so as to inform the requestor of the result of executing said local transaction.

53. [New] The process of claim 46 wherein the steps in step D of translating said Formal Language Specification to generate computer readable code which controls a computer to allow a logged on user to interact with and execute the functionality associated to a global transaction comprise generating code which controls a computer to provide a mechanism to construct the message associated to said global transaction and to execute said global transaction.

54. [New] The process of claim 53 wherein the steps in step D of translating said Formal Language Specification to generate computer readable code which controls a computer to provide a mechanism to construct said message associated to said global transaction comprises the step of generating code which controls a computer to provide mechanisms to give a value to every argument of the global transaction, said mechanisms further controlling that arguments that require a value are provided a value and that any argument whose value is provided receives a valid value.

55. [New] The process of claim 53 wherein said step D includes the steps of translating said validated formal language specification to generate computer readable code which can control a computer to execute a service which is a global transaction and comprises the following steps:

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to verify that every precondition that is defined for the global transaction being executed and for the agent class the user logged on to the system belongs to, is satisfied, and should any of said preconditions not hold, then said mechanism will produce an error with the error message being defined for said precondition that does not hold, and causing execution of said global transaction to stop and roll back all changes made to any object the state of which has been modified by execution of said global transaction;
- for every service included in a transaction formula of said global transaction being executed:

- o if said service has an associated guard, translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to check that said guard associated to said service holds, and if said guard does not hold, the rest of the steps associated to said service will be omitted;
- o translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to determine the set of objects on which said service will be executed and to provide a mechanism to execute said service on every object of said set of objects by:
  - providing mechanisms to give a value to every argument of said service, said mechanisms further ensuring that arguments that require a value are provided a value and that any argument whose value is provided receives a valid value;
  - providing a mechanism to invoke the execution of said service on said object and control the result of said execution, and should said execution result in an error, causing the execution of the global transaction to stop and roll back all changes made to the state of any object the state of which might have been modified by the execution of said global transaction;

- translating said validated formal language specification so as to provide code which can control a computer to provide a mechanism to check that every integrity constraint holds that is defined in classes whose instances include objects the state of which has been modified by the execution of said global transaction, for any object whose state has been changed by said global transaction, and should any of said integrity constraints not hold, said mechanism produces an error with an error message defined for said integrity constraint that does not hold thereby causing execution of said global transaction to stop and roll back all changes made to any object the state of which has been modified by execution of said global transaction;
- translating said validated formal language specification so as to provide code which can control a computer to provide a mechanism to check every trigger condition on any object the state of which has been modified by execution of

said global transaction, for every trigger relationship defined on each class owning any object the state of which has changed by execution of said global transaction, and, for every trigger condition that holds, a mechanism to execute the service associated to said trigger relationship by:

- providing a mechanism to determine a set of objects on which said service associated to said trigger will be executed, and
- providing a mechanism to execute said service on every object of said set of objects by:
  - giving a value to every argument of said service, and ensuring that every argument that requires a value is provided a value and that any argument whose value is provided receives a valid value; and
  - invoking the execution of said service on said object;

and

- translating said validated formal language specification so as to provide code which can control a computer to provide a mechanism so as to inform the requestor of the result of executing said global transaction.

56. [New] The process of claim 45 wherein the steps in step D of translating said formal language specification to generate computer readable code which can control a computer to allow a logged on user to interact with and execute the functionality of a query as part of filter patterns independently or in the context of a Class Population Presentation Pattern comprise generating computer code to control said computer to provide a mechanism to construct the message associated with said filter and to execute the query associated with said filter.

57. [NEW] The process of claim 56 wherein said step D includes the steps of translating said validated formal language specification to generate computer readable code which can control a computer to provide a mechanism to construct said message associated to the filter comprise generating code to control a computer to perform the following steps:

- providing a mechanism to indicate a filter whose associated query is to be executed;

- providing mechanisms to give a value to every filter variable of a filter pattern of said filter, said mechanisms further controlling that filter variables that require a value are provided a value and that any filter variable whose value is provided receives a valid value;
- providing a mechanism to indicate a display set pattern to be used in performing the query
- providing a mechanism to indicate an order criterion pattern, if any, to be used in performing said query.

58. [New] The process of claim 57 wherein said step D includes the steps of translating said validated formal language specification to generate computer readable code which can control a computer to provide a mechanism to execute the query associated with said filter and comprises the following steps:

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to access the population of instances of the class owning said filter pattern from memory;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to retrieve instances of said population which fulfil the condition stated by the filter formula of said filter depending on the values assigned to every filter variable of said filter;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to access only part of the state of every instance of said population matching said condition stated by said filter formula of said filter, said part of said state being defined said display set selected in said message, said part of said state dictated by said display set being further constrained to the attributes said user logged onto said desired computer program is allowed to query;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to return the instances of said population which fulfil said condition stated by said filter formula and, if an order criterion pattern has been indicated in said message, return said instances of said population which fulfil said condition stated by said filter formula in the order stated by said order criterion pattern.

59. [New] The process of claim 45 wherein step D comprises the steps of translating said formal language specification into computer readable code which can control a computer to provide user interface interaction mechanisms which allow a user who havs logged on to execute services and queries by interacting with one or more user interface interaction scenarios implemented as Service Presentation Patterns, Instance Presentation Patterns, Class Population Presentation Patterns and/or Master/Detail Presentation Patterns.

60. [New] The process of claim 59 wherein the steps in step D of translating said validated formal language specification to generate computer readable code which can control a computer to allow a logged on user to execute a service by interaction with a Service Presentation Pattern related to said service, said steps of translating comprising steps to generate computer readable code which implements a set of mechanisms that collaborate to construct a message for the execution of the service associated to said Service Presentation Pattern, including:

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to identify said Service Presentation Pattern by its alias;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present a help message associated with said Service Presentation Pattern to said user interacting with said Service Presentation Pattern;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to identify each argument of said service associated with said Service Presentation Pattern by the alias of each said argument of said service;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present each argument of said service associated with said Service Presentation Pattern to said user interacting with said Service Presentation Pattern in the order and groups dictated by the Arguments Grouping Presentation Pattern (if any) associated

to said Service Presentation Pattern;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to let the user provide a value for each argument of the service associated to said Service Presentation Pattern
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to ensure that every argument of said service associated with said Service Presentation Pattern that requires a value has a value and to validate that every argument that has a value has a valid value according to every said argument data type and said Introduction Pattern, if any, associated with every said argument;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present the user with the default value, if any, of every argument of said service associated to said Service Presentation Pattern;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to access an objects selection mechanism corresponding to a Population Selection Pattern, if any, associated with every object valuated argument of the service associated with said Service Presentation Pattern;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user with a help message, if any, associated to every argument of said service associated to said Service Presentation Pattern;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to implement a Dependency Pattern, if any, associated with every argument of the service associated with said Service Presentation Pattern, said mechanism:
  - o controlling a computer to monitor the occurrence of said user interface interaction events relevant to every said argument, which either change the value of said argument or activate/deactivate said argument;
  - o controlling a computer so as to check that the condition of an Event-Condition-Action (hereafter ECA) rule of said Dependency Pattern holds,

and

- executing the actions in said ECA rule of said Dependency Pattern to assign a value and/or activate and/or deactivate other arguments of the service associated to said Service Presentation Pattern;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user the elements in a Display Set Pattern, if any, assigned as a Supplementary Information Pattern to every object valuated argument of said service associated with said Service Presentation Pattern, whenever the value of every said argument changes;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to allow a user to cancel the interaction of said user with said interaction scenario represented by said Service Presentation Pattern;

- a mechanism to confirm and send the message which will cause the service associated to said Service Presentation Pattern to execute.

61. [New] The process of claim 59 wherein the steps in step D of translating said validated formal language specification to generate computer readable code which can control a computer to allow a logged on user query information on an instance of a class, execute services on said instance and/or navigate to interaction scenarios displaying information related with said instance, by interaction with an Instance Presentation Pattern of the class owning said instance, said steps of translating comprising the following steps:

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to identify said Instance Presentation Pattern by its alias;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present a help message, if any, associated with said Instance Presentation Pattern;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user with the value of each element in the Display Set Pattern of said Instance

Presentation Pattern, each element identified by the alias of said element and presented in the order dictated by said Display Set Pattern, said elements being further restricted to those elements that correspond to attributes the logged on user is allowed to query;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to access each user interface interaction scenarios corresponding to each service that can be executed on the object displayed by said Instance Presentation Pattern, each of said services identified by the alias of the Service Presentation Pattern corresponding to each of said services, said services being further restricted to the ones the logged on user is allowed to execute;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user with a help message associated with each service, if any, executable on said object displayed by said Instance Presentation Pattern;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to access each user interface interaction scenario corresponding to each class, if any, owning instances related with the instance belonging to the class owning said Instance Presentation Pattern, each of said user interface interaction scenarios being identified by the alias of its corresponding Presentation Pattern, said user interface interaction scenarios being further restricted to the ones corresponding to classes said logged on user is allowed to query;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user with any help message associated with each of the user interface interaction scenarios corresponding to each class, if any, owning instances related with instance belonging to the class owning said Instance Presentation Pattern; and
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to cancel the interaction of the logged on user with the user interface interaction scenario represented by said Instance Presentation Pattern.

50

62. [New] The process of claim 59 wherein the steps in step D of translating said validated formal language specification to generate computer readable code which can control a computer to allow a logged on user to query information on a set of instances of a class, execute services on any instance in said set and or navigate to interaction scenarios displaying information related with any said instance of said set, by interaction with a Class Population Presentation Pattern of the class owning said set of instances, said steps of translating comprising the following steps:

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to identify said Class Population Presentation Pattern by its alias;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present any help message associated with said Class Population Presentation Pattern;
- translating said validated formal language specification so as to provide code which controls a computer to provide a set of mechanisms to obtain and display a set of instances of the class owning said Class Population Presentation Pattern, comprising:
  - o a mechanism to identify the Filter Pattern, if any, associated to said Class Population Presentation Pattern;
  - o a mechanism to identify and select one of the Order Criterion Patterns, if any, associated to said Class Population Presentation Pattern;
  - o a mechanism to identify each Filter Variable, if any, of said Filter Pattern, if any, associated to said Class Population Presentation Pattern, said identification by the alias of each of said Filter Variables, and to present said Filter Variables to said user in the order they are defined in said Filter Pattern;
  - o a mechanism to present said user a default value, if any, of each Filter Variable, if any, of said Filter Pattern, if any, associated to said Class Population Presentation Pattern;
  - o a mechanism to let said user provide a value for each Filter Variable, if any, of said Filter Pattern, if any, associated to said Class Population Presentation Pattern;

- o a mechanism to validate the value assigned to each Filter Variable, if any, of said Filter Pattern, if any, associated to said Class Population Presentation Pattern, said validation carried out according to:
  - the data type of said Filter Variable;
  - what is dictated by the Introduction Pattern, if any, associated to said Filter Variable;
- o a mechanism to access the objects selection mechanism corresponding to a Population Selection Pattern which defines the process of observing and selecting objects in a multiple objects society, said Population Selection Pattern being associated to every object valuated Filter Variable, if any, of said Filter Pattern, if any, associated to said Class Population Presentation Pattern;
- o a mechanism to present said user with a help message, if any, associated to each Filter Variable, if any, of said Filter Pattern, if any, associated to said Class Population Presentation Pattern;
- o a mechanism to present said user with elements in any Display Set Pattern assigned as a Supplementary Information Pattern to every object valuated Filter Variable, if any, of said Filter Pattern, if any, associated to said Class Population Presentation Pattern, whenever the value of every said Filter Variable changes;
- o a mechanism to invoke execution of the query represented by said Filter Pattern, if any, associated with said Class Population Presentation Pattern, or to invoke the retrieval of the full population of said class owning said Class Population Presentation Pattern;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to display the value of each element, for every instance in the population of the class or for every instance returned as a result of executing said query represented by said Filter Pattern, if any, associated with said Class Population Presentation Pattern, said each element being in the Display Set Pattern associated to said Class Population Presentation Pattern, said each element the value of which is displayed being restricted to those elements corresponding to attributes the logged on user is allowed to query;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to select one of the objects or instances presented to the user by said Class Population Presentation Pattern;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to, upon selection of one of said objects presented by said Class Population Presentation Pattern, access each of said user interface interaction scenarios corresponding to each service that can be executed on said selected object, each of said services identified by the alias of the Service Presentation Pattern corresponding to each of said services, said services further restricted to the services the logged on user is allowed to execute;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user with any help message associated with each service executable on said selected object;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to, upon selection of one of said objects presented by said Class Population Presentation Pattern, access each of said user interface interaction scenarios corresponding to each class, if any, owning instances related with said selected object of said class owning said Class Population Presentation Pattern, each of said user interface interaction scenarios being identified by the alias of its corresponding Presentation Pattern, said user interface interaction scenarios being further restricted to the ones corresponding to classes said logged on user is allowed to query;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user any help message associated with each of said user interface interaction scenarios corresponding to each class, if any, owning instances related with said selected object of said class owning said Class Population Presentation Pattern; and
- translating said validated formal language specification so as to provide

code which controls a computer to provide a mechanism to cancel the interaction of the logged on user with the interface interaction scenario represented by said Class Population Presentation Pattern.

63. [NEW] The process of claim 59 wherein the steps in step D of translating said validated formal language specification to generate computer readable code which can control a computer to allow a logged on user to interact with an interaction scenario corresponding to a Master/Detail Presentation pattern so as to query information on:

- instances belonging to the class owning said Master/Detail Presentation Pattern as presented in a Presentation Pattern referred to as a Master Presentation Pattern, which is either an Instance Presentation Pattern or a Class Population Presentation Pattern owned by the same class owning said Master/Detail Presentation Pattern, and
- instances belonging to other classes related to said class owning said Master/Detail Presentation Pattern, each set of related instances as presented by a Presentation Pattern referred to as Detail Presentation Pattern, which is either an Instance Presentation Pattern, or a Class Population Presentation Pattern, or belonging to a Master/Detail Presentation Pattern owned by said class related with said class owning said Master Presentation Pattern,

said steps of translating comprising the steps of:

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to identify said Master/Detail Population Presentation Pattern by its alias;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user any help message associated with said Master/Detail Presentation Pattern;
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user the Master Presentation Pattern of said Master/Detail Presentation Pattern;
- translating said validated formal language specification so as to provide

code which controls a computer to provide a mechanism to present said user with every Detail Presentation Pattern of said Master/Detail Presentation Pattern;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to synchronize information displayed in every Detail Presentation Pattern whenever:
  - o the object displayed in the Master Presentation Pattern changes, or
  - o the selection of one of the objects displayed in the Master Presentation Pattern changes
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to cancel the interaction of the logged on user with the interaction scenario represented by said Master/Detail Presentation Pattern.

code which controls a computer to provide a mechanism to present said user with every Detail Presentation Pattern of said Master/Detail Presentation Pattern;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to synchronize information displayed in every Detail Presentation Pattern whenever:
  - o the object displayed in the Master Presentation Pattern changes, or
  - o the selection of one of the objects displayed in the Master Presentation Pattern changes
- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to cancel the interaction of the logged on user with the interaction scenario represented by said Master/Detail Presentation Pattern.

Respectfully submitted,

Dated: February 10, 2006

Ronald Craig Fish
Reg. No. 28,843
Tel 408 778 3624
FAX 408 776 0426

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail, postage prepaid, in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents , P.O. Box 1450, Alexandria, Va. 22313-1450.
on 2/10/2006
(Date of Deposit)

Ronald Craig Fish, President
Ronald Craig Fish, a Law Corporation
Reg. No. 28,843